8. Michael

Program Name: michael.java **Input File: michael.dat**

Michael has decided to invent a new game for his chess board called Quintos. In a way it is similar to Tetris, but with three major differences. Each ***"quinto"*** is made up of five squares, not four, and in the game, overlapping is allowed. Also, they don't rotate, but retain the same orientation. His idea is for the object of the game to be the person to cover the last square on the board, but he's still working out the finer points of the rules. In the meantime, Michael needs help with a program that shows how many squares have been covered after a certain number of pieces have been played.

He has decided on three basic shapes, A, B and C, and has selected one of the five squares in each shape to be the reference cell, for the purpose of data representation, with the other four cells in relative position to that reference cell.

The A style quinto is L-shaped, with the reference cell R being the top of the L, and rest of the squares in this shape:

```
R
A
AAA
```

The B style quinto has a reference cell R also in the top left position of the overall shape, and looks like this:

```
RB
BB
B
```

The C style quinto looks like an L lying in its side, like this:

```
   C
CCCR
```

To test his program so far, he has decided to use data in a certain format, with an initial single digit integer value N to represent how many pieces have been played, and then N sets of data, with each set consisting of the letter A, B or C, indicating the shape of the letter, and then two integers X and Y indicating the position of the reference cell of that piece. He wants to know how many of the 64 squares on the board have been covered so far by the pieces.

The reference cell for each piece is located at the (X,Y) position of the chess board, just as you would find in the first quadrant of a coordinate plane. The test data will indicate how many pieces have been played, followed by the shape and coordinates of each piece.

For example, the input line: **`4 A 2 5 C 5 7 B 7 6 C 8 1`**
produces a grid like this, with a total of 20 squares covered by quintos:

```
....C...
.CCCR...
......RB
.R....BB
.A....B.
.AAA....
.......C
....CCCR
```

If one more piece is played, say A 4 3, the board now looks like this, with some overlapping caused by the new piece, and only two additional squares actually covered by the new piece, for a total of 22 squares covered.

```
....C...
.CCCR...
......RB
.R....BB
.A....B.
.AAR....
...A...C
...AAACR
```

Michael - Input: Several data sets, each on one line consisting of single digit integers and the letters A, B and C. The first integer indicates how many "quintos" are represented on each line. Each quinto is represented by the letter A, B, or C, followed by 2 integers indicating the position of the reference cell of the quinto. It is guaranteed that all of the quintos represented by this data are fully on the board, with no part of any piece out of bounds.

Output: An integer value representing how many cells in the 8X8 grid are covered by the quintos for that data set.

Sample input:
```
5 A 2 5 C 5 7 B 7 6 C 8 1 A 4 3
6 A 3 7 B 3 7 C 4 7 B 5 5 C 6 5 B 5 6
6 A 4 4 C 6 2 B 4 6 B 7 3 C 8 4 A 6 4
3 A 5 5 B 5 5 C 5 5
```
Sample output:
```
22
16
21
11
```